Learning natural locomotion behaviors for humanoid robots using human bias

Learning natural locomotion behaviors for humanoid robots using human bias

Chuanyu Yang[1], Kai Yuan[2], Shuai Heng[3], Taku Komura[4], and Zhibin Li[5]

***Abstract*—This paper presents a new learning framework that leverages the knowledge from imitation learning, deep reinforcement learning, and control theories to achieve human-style locomotion that is natural, dynamic, and robust for humanoids. We proposed novel approaches to introduce human bias, i.e. motion capture data and a special Multi-Expert network structure. We used the Multi-Expert network structure to smoothly blend behavioral features, and used the augmented reward design for the task and imitation rewards. Our reward design is composable, tunable, and explainable by using fundamental concepts from conventional humanoid control. We rigorously validated and benchmarked the learning framework which consistently produced robust locomotion behaviors in various test scenarios. Further, we demonstrated the capability of learning robust and versatile policies in the presence of disturbances, such as terrain irregularities and external pushes.**

***Index Terms*—Deep Learning in Robotics and Automation, Humanoid and Bipedal Locomotion, Learning from Demonstration**

I. Introduction

HOW can one advance the autonomous capabilities of legged robots by leveraging and incorporating 50 years of research on legged locomotion [1] [2] into a new paradigm? This paper proposes a Deep Reinforcement Learning (DRL) Framework that is able to incorporate both prior research knowledge in legged locomotion and human reference motions for training the robotic gait. We investigate to what extent human inductive bias can and should be incorporated into a learning framework to *aid the exploration* while *not limiting the discovered motion*, and *generating realistic motions*. Human bias is mainly induced by incorporating imitation data, and designing a DRL framework that emphasizes on generating realistic, implementable, and energy-efficient motions.

Design choices reflecting human inductive bias for versatile motion on the real robot can be roughly categorised by the choices influencing (a) the training results, and (b) the behavior during run-time. During training, one can influence the behavior and success of the learning in the reward design, providing reference motions, designing appropriate training procedures, and selecting the appropriate network structure.

Manuscript received: September, 10, 2019; Revised December, 29, 2019; Accepted January, 22, 2020.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the EPSRC CDT in Robotics and Autonomous Systems (EP/L016834/1), Future AI and Robotics for Space (EP/R026092/1).

[1]First, Second, Forth, and Fifth Author are with the University of Edinburgh, UK. chuanyu.yang@ed.ac.uk

[2]Third Author is with the Harbin Institute of Technology, China. heng13514479054@163.com

Digital Object Identifier (DOI): see top of this page.

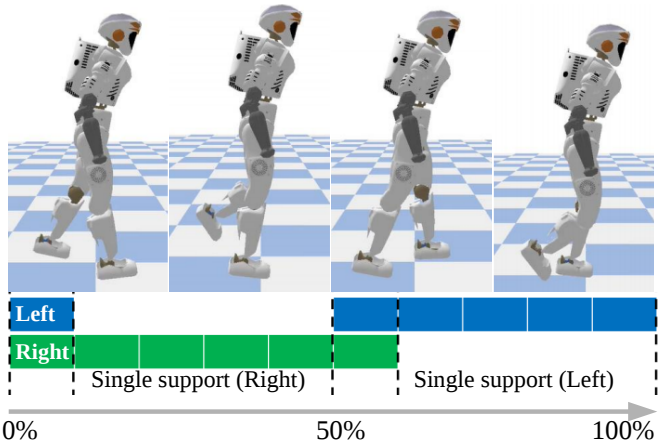

Fig. 1: Natural human-like symmetric walking pattern generated by the learning framework. The blue and green bar represents left, right foot contact phases respectively.

For run-time, the designer needs to guarantee that the trained agent produces realistic and feasible motions that can be implemented on the real system while also keeping the reality gap between simulation and the real robot as small as possible. Here, we aim to provide novel technical approaches that lead to the success of both training and deployment of learned policies on simulated robots.

The key question we want to investigate is: *how can we induce more human bias for more realistic and better motion*? We have studied three novel approaches in the framework design where bias can be introduced: 1.) initialization and termination of the state, 2.) selection of a task appropriate network structure, 3.) augmented reward design: task completion and imitation of reference motions.

The contribution of this paper are summarized as follows:

- Novel approaches to introduce human bias in the framework generating human-like, realistic motions through imitation learning;
- A multi-expert network structure with smooth blending properties for humanoid bipedal locomotion;
- Integration and design of control system and the reinforcement learning framework for better replication of the results.

In the following, related research involving leveraging human knowledge in machine learning are briefly reviewed in Section II. Background information of the robot platform, simulation environment and control framework is presented in Section III. Next, the details involved in the framework design is expanded on in Section IV. The training results are demonstrated and analyzed in Section V. Finally, the work is concluded in Section VI.

II. Related Work

Commonly, reinforcement learning attempts to automate the learning process to avoid over-using human engineering and thus preventing human bias by allowing the agent to infer the reward by itself [3]. While over-engineering and shaping the reward leave possibility of reward exploitation [4], carefully introducing human knowledge in DRL, such as specially designed network structures, can yield better results compared to baseline vanilla fully-connected neural networks [5]. Incorporating human prior knowledge into learning based methods is also known to increase the data efficiency [6]. To generate realistic motions, we review two main methods, i.e. imitating reference motions and network structure choice, in the following sections.

A. Leveraging demonstrations

By incorporating human inductive bias such as reference motions from human expert demonstrations, the quality of the resulting motions can be improved. Learning from demonstration is a technique that extracts information from the reference motion generated by expert demonstrations to guide an agent. Notable examples include Behavior cloning (BC) [7], Inverse Reinforcement Learning (IRL) [8], and Generative Adversarial Imitation Learning (GAIL) [9]. BC minimizes the difference between the student and expert behavior in a supervised learning fashion. IRL predicts a reward function such that RL can reproduce the demonstrated motion. GAIL learns a discriminator to measure the similarity between demonstrations and behaviors generated by the policy.

Directly leveraging demonstrations in the reinforcement learning paradigm can be achieved through the use of a tracking reward [10]. This method involves designing a reward dedicated to measuring the similarity between the robot state and the demonstration dataset. The tracking reward will then be combined with the task reward.

B. Leveraging human knowledge in network design

Introducing human preferences by designing special network structures has shown to improve the overall performance and learning speed of the agent in multiple benchmarking simulation environments [5].

Residual policy learning methods involves hand-designing a base policy, and learning a residual policy that augments upon the base policy to adapt to external disturbance [11]. Many specially designed network structures fall into this category. For locomotion, a base periodic trajectory is hand designed to generate periodic gaits, and a residual neural network is added upon to regulate the output of the base policy [12]. Additionally, this type of residual networks are not only used to construct policies, but also to augment simulators for more realistic dynamics [13].

Mixture of Experts (MoE) is a supervised learning architecture composed of of many separate experts, each of which is specialized for a subdomain of the task. A gating mechanism is responsible for selecting the required expert for a given input. Special neural network architectures inspired by MoE, such as Phase-Functioned Neural Networks (PFNN) [14] and Mode-Adaptive Neural Networks (MANN) [15], have been proposed to generate smooth locomotion behaviors for animation of humanoid and quadrupedal characters

Additionally, a bio-inspired approach uses the Central pattern generator (CPG), a neural oscillator, to model the functionality neural circuits within the spinal cord of verterbretes for rythmic movements. CPG is commonly used to construct the bio-inspired control policy for the locomotion of legged robots due to their ability to produce coordinated rhythmic and periodic gaits [16].

III. Learning Setup for Locomotion

In this section, we describe the specifications of the robot platform and the simulation environment, and detail the control structure and the human demonstration data.

A. Control Structure

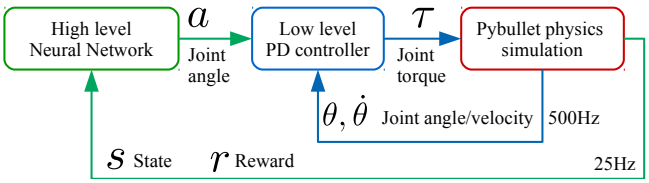

Fig. 2: The control diagram: the neural network is built using Tensorflow [17]; the simulation is built using PyBullet [18].

The control framework contains two layers. The high-level control loop consists of a DRL agent that operates at 25Hz, while the low-level control loop consists of a PD controller that operates at 500Hz. The neural network generates desired joint angles for the low-level PD controllers that produce joint torques (c.f. Fig. 2).

B. Robot platform

The algorithm is simulated in an environment with the Valkyrie robot platform. Valkyrie has a total of 26 DOF, for which only the 15 lower body joints are actuated in this paper: 3 waist joints (roll, pitch, yaw), and two 6 DoF leg joints (hip roll, hip pitch, hip yaw, knee pitch, ankle pitch, ankle roll).

1) Action space: The outputs of the policy are the 15 target joint angles for the lower body joints of Valkyrie robot. The target joint angles are sent to the PD controller to be translated into torque for the joint motor. Low level joint position control was chosen over torque control as this has been shown to achieve better performance [19].

2) State space: The policy only receives egocentric proprioceptive features as state observations. The state is adjusted to align with the gravity vector in relative coordinates, and is invariant to the yaw orientation of the pelvis in the world frame, and thus not affected by any steering in yaw orientation. The state space is chosen as in [20] and consists of joint angles and velocities, end-effector-to-pelvis vector, pelvis linear velocity and angular velocity, pelvis orientation relative to the gravity vector, Center of Mass (CoM) velocity, CoM-to-pelvis vector,

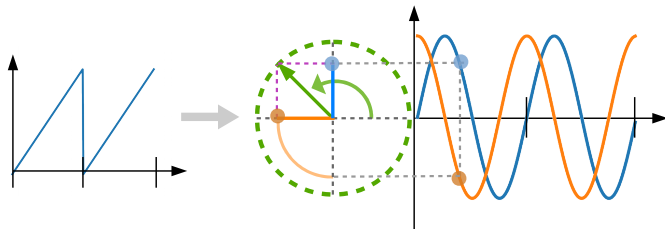

Fig. 3: The 1D Sawtooth phase is projected onto a 2D unit-cycle for a smooth transition between each cycles.

and ground contact force. The observation is filtered using a low-pass Butterworth filter.

An additional time input is added to the state to provide a time reference. Providing only reference motion for imitation learning without time will introduce temporal ambiguity, since the agent will not be able to infer the temporal relations of the reference motions. Time information is provided as normalized phase input that increments from 0 to 1 and resets after a certain time period [10]. This phase function, however, has an abrupt discrete jump from 1 to 0 that may cause non-smoothness in the learning behavior in the neural network. To mitigate this problem, we generate smoothness by projecting the 1D phase onto a 2D unit-cycle (Fig. 3).

C. Human motion Collection

The reference features from motion capture data originate from a human of different morphology than the robot. The human motion data is preprocessed before being used as reference for imitation learning. Since the human foot is narrower than the robot's, the reference hip yaw and hip roll angles are set to 0 to avoid the robot tripping on its own feet. Due to its limited influence on the gait, the torso roll and torso yaw are also set to 0.

D. Reinforcement Learning

The goal of reinforcement learning is to find an optimal policy that maximizes the discounted return. The discounted return is used as an evaluation of performance and is determined by summing the exponentially discounted reward,

$$R_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}, \tag{1}$$

where T is the total number of samples in an episode and γ is the discount factor.

1) Choosing the Discount Factor: Alternatively to tuning the discount factor γ, the half-life of discounted future rewards can be used as a reference [21], [20]. For locomotion, a time horizon of one foot step with a duration around 0.5s is enough to plan a stepping strategy. At a frequency of 25Hz this equates to 13 time steps. We choose the discount value in a way that the half-life of the future reward occurs at 0.5s, meaning that the accumulated discount factor becomes 0.5 at 13 time step $\gamma^{13} = 0.5$, hence $\gamma \approx 0.95$.

2) Policy Optimization: In this paper, we chose to use Proximal policy optimization (PPO) [22]. PPO is an on-policy DRL algorithm that tackles the problem of convergence by constraining the update step size through the use of clipped surrogate objective.

$$\begin{aligned} L^{\text{CLIP}} &= \mathbb{E}_t\left[\min(r_t(\pi_\theta)A_t, \text{clip}(r_t(\pi_\theta), 1-\epsilon, 1+\epsilon)A_t)\right] \\ r_t(\pi_\theta) &= \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, \quad \mathcal{L}_{\text{PPO}} = -L^{\text{CLIP}}. \end{aligned} \tag{2}$$

L^{CLIP} is the objective function that PPO maximizes. The term $\text{clip}(r_t(\pi_\theta), 1-\epsilon, 1+\epsilon)$ clips the probability ratio, discouraging large policy changes. Furthermore, the clipped objective $\text{clip}(r_t(\pi_\theta), 1-\epsilon, 1+\epsilon)A_t)$ is compared to the unclipped objective $r_t(\pi_\theta)A_t$, and the lower bound is chosen. The advantage A_t will be computed using generalized advantage estimator (GAE) [23].

3) Bounding the Action Space: Bounding the action space is important as the real robot systems have actuation limits. Using hyperbolic squashing functions $\tanh$ for bounding the action space have been shown to be disadvantageous due to saturation close to the boundary [24].

Instead of providing a hard constraint on the boundary of the network, we propose to implement a soft constraint by designing a bounding loss. Penalty is applied when the output of the neural network μ exceeds the lower Δ_{low} and upper Δ_{up} boundary, allowing the network to operate with outputs near the limit without saturation.

$$\mathcal{L}_{\text{bound}} = \begin{cases} 0, & \Delta_{\text{low}} \leq \mu \leq \Delta_{\text{up}} \\ 0.5(\mu - \Delta_{\text{up}})^2, & \Delta_{\text{up}} < \mu \\ 0.5(\Delta_{\text{low}} - \mu)^2, & \mu < \Delta_{\text{low}} \end{cases}. \tag{3}$$

The loss function $\mathcal{L}$ used for back-propagation is the sum of $\mathcal{L}_{\text{PPO}}$ and $\mathcal{L}_{\text{bound}}$:

$$\mathcal{L} = \mathcal{L}_{\text{PPO}} + \mathcal{L}_{\text{bound}}. \tag{4}$$

IV. Framework Design

The major contribution of this paper will be discussed thoroughly here: the approaches of introducing human bias, e.g. reference motions and specialized network design, into the learning framework through imitation reward design.

A. Reward design

Reward design is an important aspect in reinforcement learning as it governs the behavior of the agent. Our reward consists of a task term and imitation term similar to that presented in [10]. The task term provides the guidance necessary for the agent to achieve the locomotion objective, while the imitation term biases the behaviour towards a human-preferred walking pattern.

Due to their bounded nature within the range of $[0, 1]$, Radial Basis Function kernels are preferred for shaping the reward [25], [20], [21] and will be used in the following:

$$K(\hat{x}, x, \alpha) = e^{-\alpha(\hat{x}-x)^2}, \tag{5}$$

where x is the current state, and $\hat{x}$ is the desired value for that state. The hyperparameter α controls the width of the kernel.

TABLE I: Detailed description of imitation reward terms. The imitation reward terms are used to measure the distance between the generated and the reference motions.

Imitation reward terms	
Joint position	$K(\hat{\theta}_i, \theta_i, \alpha_i)$ (i represents joint index)
Ground contact	$\begin{cases} 1, & \text{matches desired foot contact} \\ 0, & \text{does not match desired foot contact.} \end{cases}$

The correct choice of α largely impacts learning as it guides the gradient of the reward and thus gives crucial signals for correct credit assignment of the DRL algorithm.

The reward $r = w_1 r_{\text{imitation}} + w_2 r_{\text{task}}$ consists of an imitation term $r_{\text{imitation}}$ and a task term r_{task} weighted by the weights w_1, w_2. The imitation term encourages the robot to follow the human demonstration, while the task term encourages the robot to satisfy the task specific objective:

1) Imitation reward: The imitation reward consists of a joint position tracking $r_{\text{joint\_pos}}$ and foot ground contact tracking r_{contact} (c.f., Table I):

$$r_{\text{imitation}} = r_{\text{joint\_pos}} + r_{\text{contact}}. \tag{6}$$

The joint position reward $r_{\text{joint\_pos}}$ is calculated at phase ϕ by measuring the difference between the measured joint angle q_ϕ and the target joint angle reference $\hat{q}_\phi$. Furthermore, the foot contact has to match the contact configuration in the motion reference, represented by the support phase in the human motion data.

2) Task reward: The task reward is the sum of multiple reward terms. Each individual reward term reflects different physical aspects of the system:

$$\begin{aligned} r_{\text{task}} = & r_{\text{pose}} + r_{\text{COM\_pos}} + r_{\text{COM\_vel}} + r_{\text{yaw\_vel}} \\ & + r_{\text{contact}} + r_{\text{fail}} + r_{\text{torque}}. \end{aligned} \tag{7}$$

The task reward for the locomotion objective is constructed using the designed principle presented in [20]. The paper presents a reward design principle for bipedal balancing tasks. By setting the target COM velocity in the x axis to positive $0.5m/s$, the reward for balancing can be repurposed for locomotion objectives. For further details of the reward design, please refer to [20].

B. Network design

Human locomotion is inherently periodic, and is therefore reasonable to incorporate structures and elements that are periodic in nature into the network design. In this work, we investigate the periodic characteristics of PFNN (Fig. 4a) and MANN (Fig. 4b) and their effects on locomotion tasks.

Considering the unique gating mechanism in MANN and PFNN, the state input of PFNN and MANN differs from FCNN. For PFNN, the phase input is isolated from the other state and is fed into a phase function. For MANN, the input S_{g} for the gating network consists of the phase, joint positions and joint velocities. The expert networks for both PFNN and MANN have access to all state features except from the phase. For FCNN, all available states including the re-parameterized phase are sent into the network.

Unlike most networks where network parameters θ stay fixed during runtime, the parameters of a PFNN are function

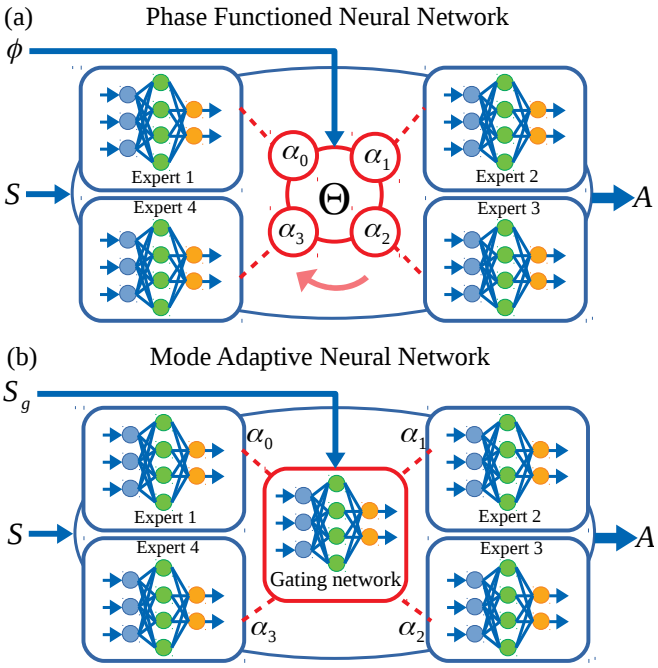

Fig. 4: The detailed structure of PFNN and MANN. Both have a gating mechanism that generates the blending weights α_i, which are used to blend the expert networks to construct the prediction network.

values that change depending on the phase variable ϕ. Within PFNN exists multiple individual sub-networks which we refer to as expert networks, each of which specializes in a particular task. The expert networks are not directly accessed, instead they are blended to reconstruct a separate network which we will refer as prediction network.

The parameters of the prediction network are computed during runtime by performing a weighted sum operation to blend the parameters of the expert networks:

$$\theta_{\text{prediction}} = \sum_{n}^{i} \alpha_i \theta^i_{\text{expert}}, \tag{8}$$

where α_i are the blending weights generated by a phase function $\Theta(\phi)$. For MANN, the phase function $\Theta(\phi)$ is replaced with a separate gating network $G(S_{\text{g}})$ to generate the blending weights α_i. For a detailed explanation of PFNN and MANN, please refer to [14], [15] respectively.

C. Sample collection

In locomotion tasks, not all states are reversible. Due to the existence of gravity, the robot is naturally inclined to fall towards the ground. The distribution of the samples will thus be biased towards samples in which the robot is struggling on the ground to get up. Those samples are not necessarily good for the network to learn to achieve the desired locomotion tasks. We thus augment the sample distribution in favor of samples that are relevant to the tasks by changing both the initialization and termination of the episode in certain states:

1) Initializing starting state: A disadvantage of fixed state initialization lies in the required time for the agent to learn to encounter high value states. Furthermore, the collected

samples will suffer from a lack of diversity since it will be dominated by states close to the fixed initialization. The following approaches have been proposed to leviate this problem:

- *Initialize from demonstration:* In cases when demonstrations are available, it is common to initialize the agent with a state sampled from demonstration trajectory [10]. In the absence of demonstration states, hand-designed reference states can also be added [21].
- *Initialize from history samples:* Alternative to initializing from external demonstrations, selected, past samples as initialization states is applied [26].
- *Learning separate initialization policy:* A separate reset policy can be trained in parallel to the task policy. This method is necessary for learning in real world where instant reset is impossible [27].

2) Early termination: During early termination, the rollout is terminated when the robot reaches a terminal state. The terminal state can be either a successful goal state or an irreversible fail state. Early termination diversifies the sample distribution by increasing the reset frequency. For scenarios with irreversible fails states, termination and resetting the rollout prevents the samples from being dominated by fail states during the early learning stage:

- *Termination with physical criteria:* A common way to determine a irrecoverable fail state is to use a physical criteria from the environment, i.e. pelvis height and undesirable body contact for locomotion tasks [10].
- *Termination with Critic value:* A low critic value can be used as a makeshift criteria to determine a fail state [27] as fail states tend to have low reward, which would then be reflected in the learned critic value.
- *Early termination due to time constraint:* Even in cases where there are no irreversible fail state, it might be useful to terminate the episode after a prolonged period for more frequent reset to diversify the samples [28].

Reference state initialization can be combined with early termination to augment the sample distribution in a way that increases sample diversity. In our work, we initialized the robot using joint references from human motion capture data. The termination condition is triggered when the pelvis height is beneath 0.5m and when the upper body contacts with the ground. The episode will be also be terminated and reset after a prolonged period exceeding the time constraint of 30s.

V. Results

We first present the learning results compared to the achieved cumulative, undiscounted reward, and then quantitatively and qualitatively show the effect of imitation learning and the choice of the network structure in a comprehensive performance analysis based on three criteria: stability, robustness, and energy consumption.

A. Learning and Comparison Setup

All agents are trained on a commercial Intel i9 CPU equipped with a Nvidia 2080Ti GPU. For comparison purposes, each policy is trained for 400 iterations with 4096 steps

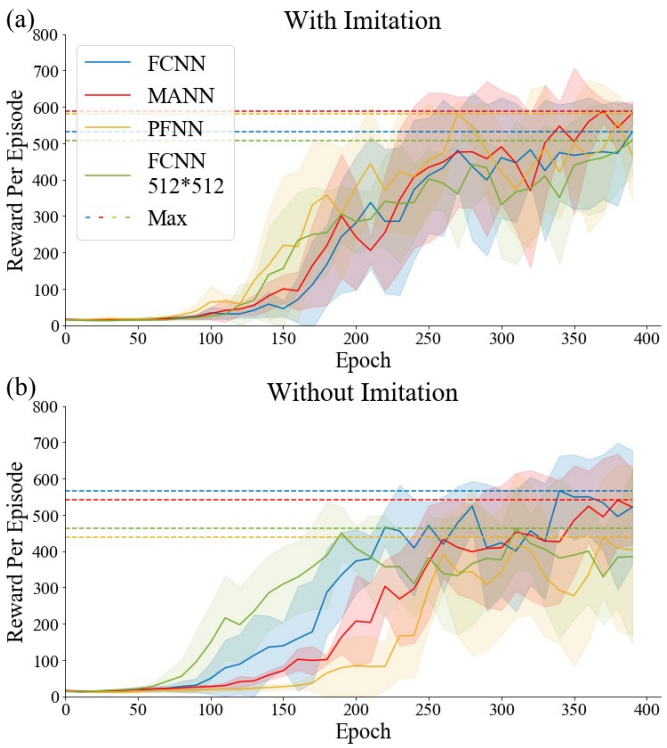

Fig. 5: Learning curve for 4 different network setups averaged over 5 trials.

for each iteration using either a FCNN, PFNN, or MANN network structure and converges after 48 hours. Each network consists of two hidden layers, the multi-expert networks use 4 experts with 128×128 neurons. MANN has an extra gating network with 32×32 neurons. To guarantee that the behaviors and performances originate from the network structure and is not constrained by the amount of neurons available, we trained with larger amounts of neurons.

At each iteration the cumulative reward is obtained by executing the deterministic policy on the current state. The reward for including imitation is calculated by $r = 0.5 \cdot r_{\text{imitation}} + 0.5 \cdot r_{\text{task}}$; and $r = 0.0 \cdot r_{\text{imitation}} + 1.0 \cdot r_{\text{task}}$ if no imitation is used (c.f. Section IV-A). At a maximum reward of 1 in a single timestep, the highest achievable cumulative reward for a complete episode is 750 at 750 time steps (30s).

Fig. 5 depicts the learning curves for different network structures averaged over 5 trials. With imitation reward provided, MANN obtains the highest reward followed by PFNN in close margins. Without imitation, the FCNN with 128 neurons (FCNN 128×128) has the highest reward.

B. Analysis of the Influence of Imitation Learning

The agent is able to learn a successful policy both with and without imitation reward. The task reward itself is sufficient to generate a locomotion behavior. However, the most significant difference arises in the gait pattern. By including the imitation reward, the agent is able to learn a symmetric gait that resembles the human walking motion data from which it learns. Without imitation learning the agent learns an asymmetric leaping gait with one leg constantly in the front and the other at the back (c.f. Fig. 7b).

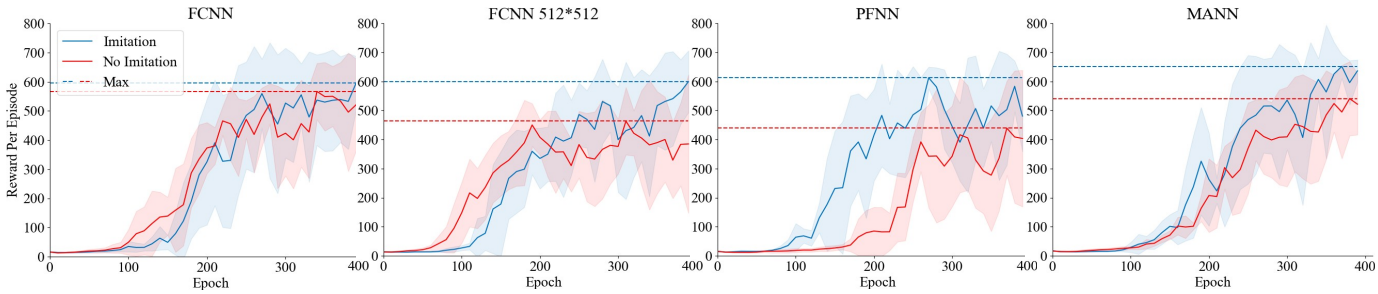

Fig. 6: Learning curve of the task reward component. The addition of the imitation reward allows the agent to achieve the task objective much better, reflected by the higher task reward value.

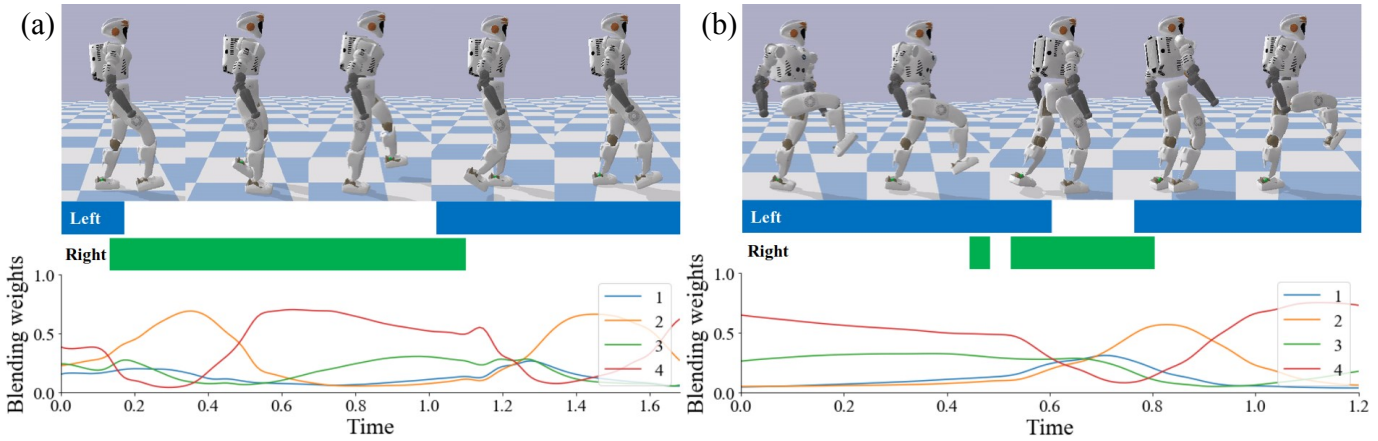

Fig. 7: Locomotion behavior of MANN (a) with and (b) without imitation. The blending weights fluctuate over a gait cycle, indicating that the primitive networks are activated differently during different gait phases, demonstrating specialization within the multi-expert structure. (a) a human-like gait pattern with symmetrically distributed left, right foot contact periods. (b) an un-human like gait pattern with asymmetric foot contact period.

TABLE II: Maximum reward during each episode. MANN with imitation achieves the highest task reward.

Neural Network	w/ Imitation		w/o Imitation
	Imitation term	Task term	Task term
FCNN 128×128	465	596	567
PFNN	549	612	439
MANN	528	**652**	541
FCNN 512×512	416	599	463

Introducing the imitation term not only creates a better, human-like gait pattern, but also improves the overall locomotion task performance of the policy. From Fig. 6 and Table II we can see that including the imitation term allows the agent to learn a policy that achieves a higher reward in the task term, meaning that adding human demonstration also allows the agent to achieve the locomotion task better. Amongst all combinations, MANN with imitation achieves the best performance with respect to the collected reward.

However, introducing an imitation term reduces the performance of the agent in terms of generalization (tasks and environments that the policy was not trained in), such as disturbances and walking over uneven terrain (c.f. Table III). This decline in performance can be explained by the fact that the agent encounters less distinct states as the imitation reward encourages the agent to be in states as close as possible to the reference motion.

C. Comparison Study between Neural Network Structures

The PFNN and MANN network consist of 4 individual experts with two hidden layers containing 128 neurons, therefore having 4 times the neurons. For fair comparison, we included another FCNN with 512 neurons in the hidden layer to even out the advantage of PFNN and MANN.

FCNN (512×512) performs poorly on locomotion in both with and without imitation learning scenarios even though having the same amount of neurons with PFNN and MANN, and 4 times the neurons of FCNN (128×128). Showing that the behavior is not limited by the expressiveness of the network and increasing network size does not guarantee improvement. This further indicates that the high performance from MANN and PFNN is a result of the network structure rather than the increased number of neurons.

From Fig. 5, a difference for all three neural networks in convergence speed and the converged cumulative reward can be seen. With imitation learning, MANN and PFNN converges to a higher reward with faster speed compared to that of FCNN. Furthermore, it can be seen that PFNN performs poorly on the locomotion tasks without imitation reference provided. However, PFNN is able to generate human-like symmetric periodic gaits without human reference due to its inherent periodic structure, unlike FCNN and MANN which relies on

the human demonstration.

We recorded the output of the gating network of MANN to analyze the specialization of each experts. The blending weights α_i for each expert show a distinctive pattern that corresponds to the phase cycle of the walking gait (Fig. 7), which indicates that the MANN has an understanding that specialization is necessary for different instance in the phase.

Many works have used Genetic Algorithm (GA) to obtain the parameters for CPG [29], while some others have used policy search reinforcement learning [30]. However, training CPG with policy search requires special modification of the RL framework [30]. Also, due to its internal states, the CPG neuron is regarded as one type of recurrent neural network (RNN), thus training CPG with policy search requires Back-Propagation Through Time (BPTT). Therefore, CPG has not been included in our comparison study as it requires a different technique - BPTT - in order to be compatible with standard RL procedure. In contrast, both PFNN and MANN can be trained on the same basis using the same back-propagation technique of regular FCNN.

D. Performance Comparison

In the following, we analyze the performance for using imitation, and different network structures. The performance is evaluated based on the robustness, stability, and energy efficiency of the resulting gait.

1) Robustness: Different test scenarios are used to evaluate the robustness of the learned locomotion policy:

- *Push on pelvis:* We applied various amount of forces on the robot pelvis and observe how much disturbance the robot can withstand. The robot is able to withstand a impulse up to 550Ns (5500N over 0.1s), c.f., Fig. 8(a). A comprehensive comparison across all combinations is shown in Table III. The policies that are trained without the imitation term are able to resist larger disturbances.
- *Blindly traversing uneven terrain:* Even though the agent is trained without external visual input and in a flat environment, some of the learned policy are able to generate a robust locomotion behavior traversing over uneven terrain without falling (Fig. 8(b)).
- *Persistent small disturbance:* Lastly, we investigate how the policy behaves under frequent small disturbances. Random cubes of various weights are thrown towards the robot with an initial velocity of 20m/s (Fig. 8(c)). All policies are able to withstand disturbances induced by the cubes. The robot is able to withstand heavier cubes with policies trained without imitation.

2) Stability: To investigate the stability, the Capture Point (CP) is analyzed with respect to their shortest distance to the edge of the Support Polygon (SP), which are listed in Table IV. The larger the distance, the more stable it is as the CP have a larger margin before shifting out of the SP

3) Energy Consumption: Energy consumption is an important aspect to be considered in reality, and hence, we are interested in whether including imitation reference or changing network structure influences the energy-efficiency of learned gaits. We analyze the *cost of transport* (energy consumed per distance traveled) for each policy, and found no significant correlation between the cost of transport and the training setups.

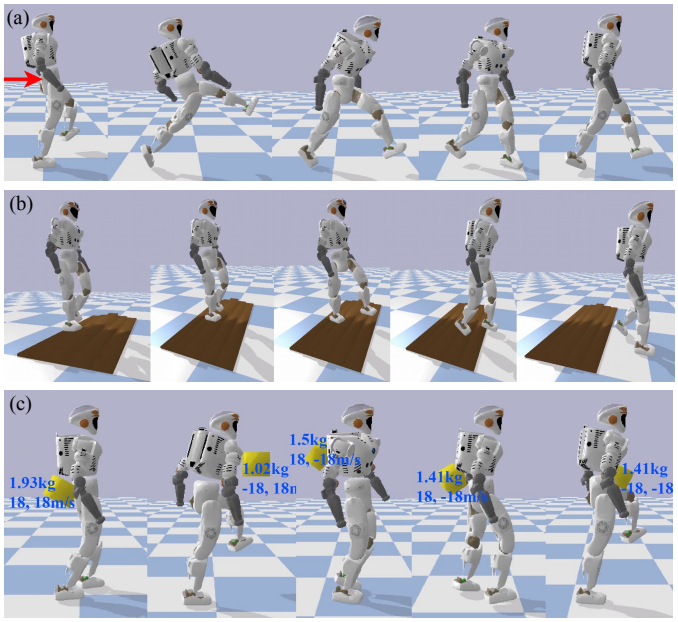

Fig. 8: Robustness: (a) 550Ns impulse on pelvis; (b) walking over stairs with variable heights: 2.5cm, 5cm, 10cm, 5cm, 2.5cm; (c) constantly throwing cubes at 20m/s initial velocity.

4) Physics simulation setting: The pybullet physics engine uses sequential impact solver to calculate the contact dynamics [18]. We set joint angle, velocity, and torque limits in the physics simulation the same as the real Valkyrie robot, so as to enforce the policy to learn motions that does not violate the physical constraints. Table V compares the peak torques and velocities from different policies. The ankle joint velocity occasionally exceeds the limit of 11.00 rad/s due to the large ground impacts.

VI. Conclusion

In this work, we present key novel design approaches of a Deep Reinforcement Learning (DRL) Framework, and demonstrate that DRL is able to learn a robust, human-like walking policies that are reactive and robust against external disturbances. In particular, the diversity of behaviors and the variety of gait patterns exhibited during different test scenarios are all learned and emerged naturally, instead of being explicitly programmed as in the traditional approaches.

The comprehensive analysis on the influence of different network structures and imitation of human demonstrations shows that every structure has its advantage: if designed properly using the framework in this paper, FCNN provides a good baseline that is able to generalize well and withstand large disturbances due to the intentionally designed training. If no reference for imitation learning is available, PFNN is able to generate periodic, symmetric gaits due to its inherent periodic structure. If a reference is available, MANN using imitation learning is able to accomplish the best task performance. For all network structures, introducing human demonstration

TABLE III: Performance analysis for imitation learning and different network structures.

Disturbance type	with Imitation				without Imitation			
	FCNN	MANN	PFNN	FCNN 512	FCNN	MANN	PFNN	FCNN 512
Impulse in [Ns]	280	280	300	290	470	420	400	**550**
Can blindly walk over stairs	✓	✓	✗	✗	✓	✓	✗	✓
Thrown cube weight in [kg]	5	4	6	7	10	7	7	**11**

TABLE IV: Performance analysis for imitation learning and different network structures.

	with Imitation								without Imitation							
	FCNN		MANN		PFNN		FCNN 512		FCNN		MANN		PFNN		FCNN 512	
Performance metric	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Distance CP to SP	0.141	0.083	0.154	0.099	0.193	0.131	0.147	0.089	0.182	0.104	0.233	0.106	0.248	0.156	**0.250**	0.116
Cost of Transport [W/m]	301	142	**258**	153	278	130	281	146	259	168	309	155	261	128	405	268

TABLE V: Peak torques and velocities of leg joints. Torso joints are omitted due to their limited influence on walking.

	Peak joint torque $[N]$						Peak joint velocity $[rad/s]$					
	Hip roll	Hip pitch	Hip yaw	Knee pitch	Ankle Pitch	Ankle Roll	Hip roll	Hip pitch	Hip yaw	Knee pitch	Ankle Pitch	Ankle Roll
Joint limit	350	350	190	350	205	205	6.11	6.11	5.89	11.00	11.00	11.00
FCNN 512	343	350	190	350	205	205	2.99	6.11	3.09	9.58	12.20	11.96
FCNN	234	350	158	350	205	205	2.40	4.69	2.75	6.61	7.01	11.00
PFNN	243	350	141	350	205	205	2.64	6.11	3.92	6.27	11.93	11.14
MANN	278	350	131	350	205	205	1.97	5.47	4.83	8.41	9.08	11.37

proves to be beneficial for locomotion tasks, which is reflected by the increased value of the task objective.

Though we have applied realistic joint torque and velocity constraints in the simulation, the learned control policy will have difficulty to be directly transferred on a real system due to the discrepancies between simulation and the real world, such as variations in mass distribution, friction, and contact dynamics etc. For future work, we plan to conduct research on the topic of simulation to reality policy transfer by bridging the gap between simulated dynamics and real world physics, in order to implement the learned policies on a real robot. Furthermore, we will also continue research on MANN for multi-task scenarios to fully exploit the potential of the multi-expert network structures.

REFERENCES

[1] Z. Li, *et al.*, "Humanoid balancing behavior featured by underactuated foot motion," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 298–312, 2017.

[2] Z. Li, *et al.*, "Walking trajectory generation for humanoid robots with compliant joints: Experimentation with coman humanoid," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 836–841.

[3] J. Fu, *et al.*, "Variational inverse control with events: A general framework for data-driven reward definition," in *NIPS*, 2018.

[4] D. Amodei, *et al.*, "Concrete problems in AI safety," *arXiv:1606.06565*, 2016.

[5] M. Srouji, *et al.*, "Structured control nets for deep reinforcement learning," *arXiv preprint arXiv:1802.08311*, 2018.

[6] K. Chatzilygeroudis, *et al.*, "A survey on policy search algorithms for learning robot controllers in a handful of trials," *arXiv preprint arXiv:1807.02303*, 2018.

[7] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, 1991.

[8] A. Y. Ng, *et al.*, "Algorithms for inverse reinforcement learning." in *ICML*, 2000.

[9] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NIPS*, 2016.

[10] X. B. Peng, *et al.*, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, 2018.

[11] T. Johannink, *et al.*, "Residual reinforcement learning for robot control," in *ICRA*, 2019.

[12] A. Iscen, *et al.*, "Policies modulating trajectory generators," in *CoRL*, 2018.

[13] A. Ajay, *et al.*, "Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3066–3073.

[14] D. Holden, *et al.*, "Phase-functioned neural networks for character control," *ACM Trans. Graph.*, 2017.

[15] H. Zhang, *et al.*, "Mode-adaptive neural networks for quadruped motion control," *ACM Trans. Graph.*, 2018.

[16] L. Righetti and A. J. Ijspeert, "Programmable central pattern generators: an application to biped locomotion control," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1585–1590.

[17] M. Abadi, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," https://www.tensorflow.org/, 2015.

[18] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[19] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?" in *ACM SIGGRAPH*, 2017.

[20] C. Yang, *et al.*, "Learning whole-body motor skills for humanoids," in *Humanoids*, 2018.

[21] J. Hwangbo, *et al.*, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, 2019.

[22] J. Schulman, *et al.*, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[23] J. Schulman, *et al.*, "High-dimensional continuous control using generalized advantage estimation," *arXiv:1506.02438*, 2015.

[24] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," *arXiv:1511.04143*, 2015.

[25] C. Yang, *et al.*, "Emergence of human-comparable balancing behaviours by deep reinforcement learning," in *Humanoids*, 2017.

[26] A. Tavakoli, *et al.*, "Prioritizing starting states for reinforcement learning," *arXiv preprint arXiv:1811.11298*, 2018.

[27] B. Eysenbach, *et al.*, "Leave no trace: Learning to reset for safe and autonomous reinforcement learning," in *ICLR*, 2018.

[28] F. Pardo, *et al.*, "Time limits in reinforcement learning," in *ICML*, 2018.

[29] N. Ogihara and N. Yamazaki, "Generation of human bipedal locomotion by a bio-mimetic neuro-musculo-skeletal model," *Biological cybernetics*, vol. 84, no. 1, pp. 1–11, 2001.

[30] T. Mori, *et al.*, "Reinforcement learning for cpg-driven biped robot," in *AAAI*, vol. 4, 2004, pp. 623–630.